doi: https://doi.org/10.2514/6.2020-1640

Deep Gaussian Process Enabled Surrogate Models for Aerodynamic Flows

Dushhyanth Rajaram*, Tejas G. Puranik†
Aerospace Systems Design Laboratory, Georgia Institute of Technology, Atlanta, GA, 30332, U.S.A.

S. Ashwin Renganathan‡
Argonne National Laboratory, Lemont, IL, 60439, U.S.A.

WoongJe Sung†, Olivia Pinon Fischer§, Dimitri N. Mavris¶
Aerospace Systems Design Laboratory, Georgia Institute of Technology, Atlanta, GA, 30332, U.S.A.

Arun Ramamurthy‖
Siemens Corporate Technology, Princeton, NJ, 08540, U.S.A.

Deep Gaussian process (DGP) models are multi-layered hierarchical generalizations of the well-known Gaussian process (GP) models widely used to construct surrogate models of aerodynamic quantities of interest. Combining the desirable features of GP models and deep neural networks (DNN), DGP models are known to perform well when training data is scarce and the behavior of the system response is highly non-stationary. In this paper, the performance of DGP models is evaluated against GP models. Detailed comparisons are made and conclusions are drawn in terms of training time, data requirements, predictive error, and robustness to choice of training design of experiments, among other metrics. Additionally, sensitivity and scalability analyses are conducted for the DGP models to evaluate their usability. Finally, an adaptive construction of both models is presented, where the models are built sequentially by selecting points that maximize posterior variance. Several experiments are conducted with canonical test functions at varying input dimensions and a viscous transonic airfoil test case at 42 input dimensions. The experiments suggest that DGP models outperform traditional GP models in terms of accuracy but incur higher computational costs for training.

I. Introduction

The design of complex engineering systems, from conceptualization to commercialization, is expensive. For example, systems such as aircraft/rotorcraft/spacecraft can take 15 to 30 years and cost billions of U.S. dollars to design and manufacture. Most of the cost is associated with building a prototype and testing it in a physical experimental setup. Under this paradigm, most of the knowledge about the product is gained after testing the prototype, at which point making any design changes costs time and money. Another consequence of such practice is that it limits the number of alternatives that can be evaluated throughout the design process. The advent and growth in supercomputing has helped mitigate this problem to some extent by allowing one to simulate the system's physics using high-fidelity (HF) mathematical models. Such models are typically based on conservation laws and solve a coupled non-linear system of partial differential equations on a discretized spatio-temporal domain. While HF models may offer a cheaper alternative to physical testing, many challenges and limitations as to their use remain to be addressed. For instance, models that capture the complex flight physics accurately typically take several days to solve one design configuration, even on a supercomputer. Since exploring the high-dimensional design space could involve 1000s of such simulations, their use in such a context is currently not feasible. A paradigm shift in the existing design process is necessary, that if successful, could result in cheaper, faster and hence more efficient engineering design cycles. A common solution is to replace the

*Graduate Research Assistant, Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, AIAA Student Member
†Research Engineer II, Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, AIAA Member
‡Postdoctoral Fellow, Mathematics and Computer Science Division, Argonne National Laboratory, AIAA Member
§Senior Research Engineer, Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, AIAA Senior Member
¶S.P. Langley NIA Distinguished Regents Professor and Director of Aerospace Systems Design Laboratory, Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, AIAA Fellow.
‖Research Scientist, Siemens Corporate Technology

actual model with a computationally cheaper, simplified model called a *surrogate* model. Ideally, a surrogate model trades a relatively small amount of accuracy for significantly larger gains in computational efficiency thereby enabling reliable, real-time decision making during early stages of aerospace design.

This paper focuses specifically on constructing surrogate models for aerodynamic quantities of interest (QoI) such as pressure, lift, drag and moment coefficients. These quantities capture the aerodynamic performance of an aircraft and are dependent on the operating flight conditions and the aerodynamic shape of the aircraft outer mold line (OML) that typically requires high-dimensional parameterization. An efficient surrogate model in this context enables one to design the optimal OML for a range of flight conditions. However, the construction of efficient surrogate models for aerodynamic QoI face several challenges. First, the behavior of the system response can be highly non-linear and display localized variability. Second, queries to the HF model to generate training data are expensive, hence limiting the size of the training dataset. Finally, the appropriate design of experiments (DOE) to generate the training data is hard to determine apriori. Given these challenges, there is a need for an approach that is robust to training dataset size and DOE, in addition to having flexibility to approximate a wide range of input-output relationships.

This work evaluates the performance of deep Gaussian process (DGP) [1] models as a potential alternative for surrogate modeling in aerospace design due to their promise in overcoming the aforementioned challenges. DGP models are hierarchical, multi-layered generalizations of Gaussian process (GP) models and have a similar architecture as deep neural networks (DNN). Therefore they share the stochastic properties of GPs, while also allowing composing them across layers similar to DNNs. The suitability of DGPs for non-stationary responses has been demonstrated by Damianou et al [1, 2] and Vafa [3] with both synthetic and real datasets. There has also been recent efforts [4] to approximate non-stationary responses using clustering and local gaussian process regressions. Bui et al. [5] showed that DGPs can outperform competing methods (GPs and Bayesian NNs) on select regression problems. DNN performs very well when large amount of data is available. However, data in aerospace design is typically generated from expensive experiments and hence are typically limited in their size. Therefore, surrogate modeling techniques that are robust with small data and flexible in approximating a wide-range of response behaviors are needed. Salimbeni and Deisenroth [6] have demonstrated on sample regression and classification problems that adding layers (and thereby complexity) to the DGP model did not result in overfitting, even with small and medium sized datasets. The only known application of DGP to aerospace applications is by Hebbal et al. [7], where they use a multi-objective expected improvement criterion for Bayesian optimization in rocket booster design. The application and benefits of DGPs over more traditional aerodynamic surrogate modeling techniques have not been investigated before. To address this gap this work compares the performance of DGPs against regular GPs and focuses on evaluating training time, training dataset size requirement, and model accuracy among other metrics. Additionally, the scalability of DGPs in higher dimensions and its sensitivity to hyperparameters are also evaluated in this paper for which limited work currently exists in the literature to the best of our knowledge. As such, this paper represents the first effort to evaluate DGPs as a surrogate model for aerospace design problems involving high-dimensional input parametrization and finite computational resources. The proposed approach is benchmarked for performance and training time against GP models.

The rest of the paper is organized as follows. The background material and theoretical basis of GP and DGP models are reviewed in section II. The experimental test case and associated parametrization is presented in section III. The results are presented on canonical test functions followed by the airfoil test cases in section IV. The paper finally concludes with a summary of the results and an outlook on future work.

II. Overview of Gaussian Process and Deep Gaussian Process Models

This section provides an overview of the theory behind GPs and DGP and insight into some challenges associated with the estimation of DGP model parameters.

A. Gaussian Process Regression

Let the inputs and observations of a physical/computer experiment be $\mathbf{x} \in \mathbb{R}^p$ and $y(\mathbf{x}) \in \mathbb{R}$, respectively. Then a GP model assumes that

$$y(\mathbf{x}) = f(\mathbf{x}) + z(\mathbf{x}) + \epsilon, \tag{1}$$

where $z(\mathbf{x}) \sim \mathcal{GP}(0, \sigma^2 r(\cdot))$ is a *zero-mean* GP with constant variance σ^2 and correlation structure $r(\cdot)$, and ϵ is assumed to be a Gaussian white noise. A parametrized function specifying the correlation structure is referred to as the *kernel* function denoted by $k(\cdot, \cdot; \theta)$), where θ are the parameters; we use $r()$ and $k()$ interchangably through the rest of the

paper. The unknown (deterministic) function $f(\mathbf{x})$ can be specified as a parametrized function but a common approach is to integrate it out to form the marginal likelihood distribution given as

$$p(y|\mathbf{x}) = \int p(y|f(\mathbf{x}), \mathbf{x})p(f(\mathbf{x}))d\mathbf{f}, \tag{2}$$

The conjugacy property of GP models leads to a closed-form expression for the marginal likelihood which makes it amenable to be estimated with conventional optimization techniques. However, this is not possible in DGP models as will be shown in section II.B. One of the key ingredients of GP models is the choice of the kernel; this work uses the radial basis function (RBF) kernel.

B. Deep Gaussian Process Regression

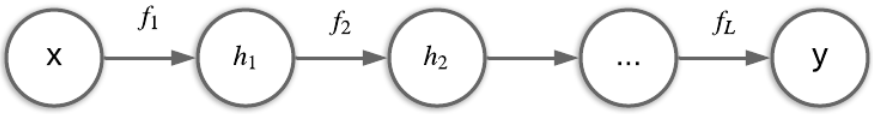


Fig. 1 Schematic of a Deep Gaussian Process (DGP) network

A schematic representation of a DGP model is shown in Figure 1 with $L-1$ hidden layers and 1 hidden unit per layer. h_i is the ith hidden layer and is mapped to hidden layer h_{i-1} via a latent function f_i, which is a GP. In other words, each hidden layer is a composition of a GP that maps it to the previous hidden layer thereby establishing the relationship between the inputs and outputs as

$$y(\mathbf{x}) = f_L\left(f_{L-1}\left(\ldots f_1(\mathbf{x})\right)\right) + \epsilon, \tag{3}$$

where each $f_i \sim \mathcal{GP}(\mu_i, \sigma_i^2 r(\cdot,\cdot;\theta_i))$ with parameters $\{\mu_i, \sigma_i^2, \theta_i\}$ and ϵ is again Gaussian white noise noise. The marginal likelihood for the DGP is obtained by integrating out the hidden layers. This is then given by the following equation

$$p(y|\mathbf{x}) = \int p(y|h_L)p(h_L|h_{L-1})\ldots p(h_1|\mathbf{x})d\mathbf{h}_1\ldots d\mathbf{h}_{L-1} \tag{4}$$

The integral in (4) does not have a closed-form expression similar to GP due to the non-linear dependence of the kernels of the probability densities on the hidden layers. Additionally, the numerical approximation can be intractable as the number of hidden layers/units increase. This is the fundamental challenge associated with the estimation of DGP models compared to regular GP models. Practical estimation methods for DGP include variational inference (VI) [8] or statistical sampling such as the Markov Chain Monte Carlo (MCMC) method [9].

III. Canonical Problems and Test Case

A. Canonical Problems

For initial testing in Section IV, the Himmelblau and Branin two-dimensional functions are utilized whereas the Ackley and Trid functions is used to demonstrate the scalability of DGP models for higher dimensions. While they are commonly available and frequently used for testing surrogate model accuracy, the descriptions and definitions from the Simon Fraser University's Virtual Library of Simulation Experiments * are used in this work. Details about these test functions are provided in Appendix V as well. The OTL circuit function is used as a candidate engineering canonical problem for testing static DOE cases and adaptive sampling. The OTL Circuit function models an output transformerless push-pull circuit. The response V_m is the midpoint voltage. The input consists of six dimensions. The input variables are $[R_{b1}, R_{b2}, R_f, R_{c1}, R_{c2}, \beta]$ and their usual input ranges are given by the following bounds: lower

*https://www.sfu.ca/~ssurjano/optimization.html

bound = [50, 25, 0.5, 1.2, 0.25, 50], upper bound = [150, 70, 3, 2.5, 1.2, 300]. The analytical form of this function is given by the following equation:

$$V_m(\mathbf{x}) = \frac{(V_{b1} + 0.74)\,\beta\,(R_{c2} + 9)}{\beta(R_{c2} + 9) + R_f} + \frac{11.35R_f}{\beta(R_{c2} + 9) + R_f} + \frac{0.74R_f\,\beta\,(R_{c2} + 9)}{(\beta(R_{c2} + 9) + R_f)R_{c1}} \ldots V_{b1} = \frac{12R_{b2}}{R_{b1} + R_{b2}} \tag{5}$$

B. Test Case and Parameterization

One of the main goals of this work is to construct surrogate models of the relationship between aerodynamic QoI, the operating flight conditions and the OML shape parameters of the aerodynamic object. Doing so first requires parameterizing the airfoil shape which is non-trivial, specifically due to regions such as the rounded leading edge (that could lead to infinite slope) and the sharp trailing edge (which could lead to a discontinuity). Consequently, parameterization that can accurately approximate any airfoil shape while also keeping the number of parameters minimal and exhibiting smooth variation to parameter changes is desired.

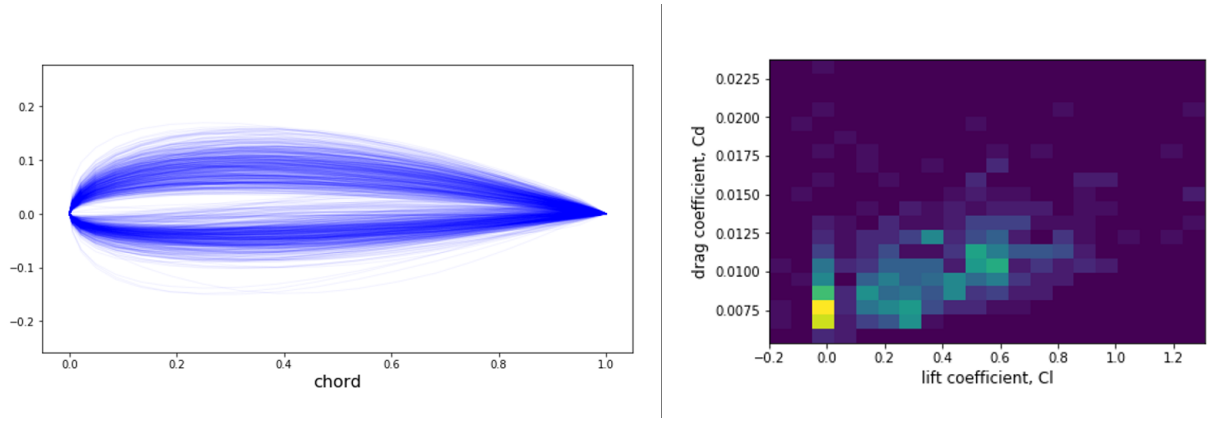


Fig. 2 Variation of airfoil shape (left) and lift/drag coefficients (right) in data set (M = 0.75, Re = 10^6, AoA = 0^o)

For scenarios with higher number of input design parameters, a 2-D airfoil shape design test problem is chosen. In this problem, the shape of the airfoil is parameterized by the mean camber position and the thickness at 22 longitudinal positions. This information is used to define total 42 points on an airfoil; 20 upper, 20 lower, leading edge and trailing edge points. The 22 fixed chordwise coordinates were clustered near both leading and trailing edges. For each airfoil shape, the non-dimensional lift and drag coefficients are calculated using XFOIL [10]. A total of over 600 airfoil shapes are evaluated at Mach number 0.75, Reynolds number 6×10^6 and angle of attack 0^o. The non-dimensional lift and drag coefficients are available to train and tune the surrogate models. The total variation of the airfoil among all the 600 shapes used in this work is shown in Figure 2 with the corresponding variations in lift and drag coefficients. It is noted that there are other methods that can also be used to parameterize airfoils such as those by Ghosh et al. [11], or class shape transofrmations (CST) [12, 13], A similar setup has been used in previous work [14, 15] for fitting a neural network models for predicting airfoil lift coefficient and other properties.

IV. Implementation and Results

In this section, the results obtained from the application of DGP to various canonical problems using a static design of experiments is demonstrated. Further, the comparison of GP and DGP in an adaptive sampling context is provided for canonical engineering problems and the aerodynamic test case outlined in section III. In each case, the comparison is made between DGP and traditional GP using the same set of training points and the same kernel function. Additionally, the computational cost in the form of training time (wall time) is also compared for both sets of models. Figure 3 shows the progression of experiments carried out in the evaluation of DGP as candidate surrogate model for high-dimensional problems of interest.

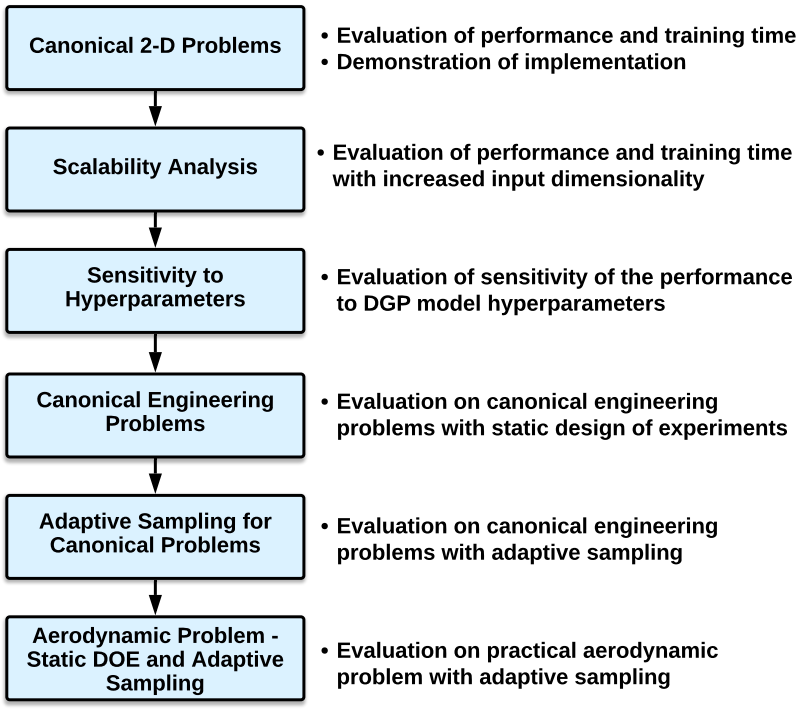


Fig. 3 Progression of experiments for DGP evaluation demonstrated in the paper

While Deep GP is a relatively newer surrogate model, there are various implementations publicly available. Among the most popular implementations is PyDeepGP from the Sheffield Machine Learning Group[†]. This implementation contains three main hyperparameters – the number of hidden layers, the number of hidden units (called latent units) per layer, and the number of inducing points. Unlike DNN models or other models with deep architectures, typically the number of hidden layers vary from 1 to 5. The number of latent units also vary from 1 to 5 in previous implementations [1, 3, 5]. The number of inducing points affects the speed of the implementation significantly and there is no clear guidance as to the appropriate number of inducing points [6]. Through grid search and trial and error on the bounds, the number of inducing points in this work is varied from 25% to 200% of the size of the training data set whereas the number of layers and latent units is varied between [1, 2]. The following sections describe the results obtained from these experiments and their implication on the utility of DGP models.

A. Canonical 2-D Problems

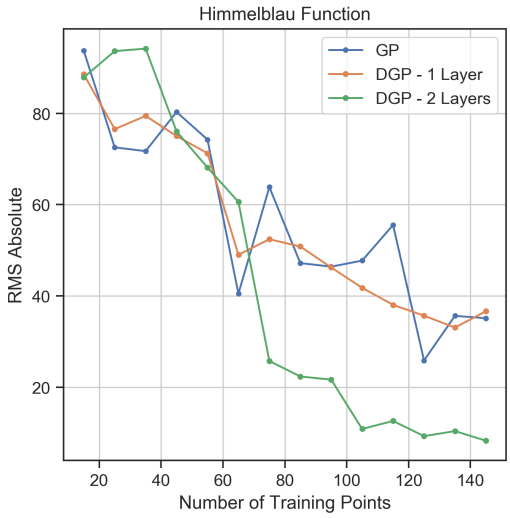
The first set of experiments conducted consider the application of DGP on popular regression benchmarks for 2-D functions. The Himmelblau and Branin test functions are used in this work. For each of these functions, GP and DGP (1-layer and 2-layer variants) models are trained using increasing number of training points (each obtained using a static latin hypercube design of experiments). In each case, the trained models are tested on a set of one hundred test points the absolute root mean square error is obtained. Both the models are trained multiple times and the errors are averaged over all the runs to ensure the effect of stochasticity is accounted for. The computational cost in terms of the wall time required for training the models is also measured for comparison purposes.

1. Himmelblau Function

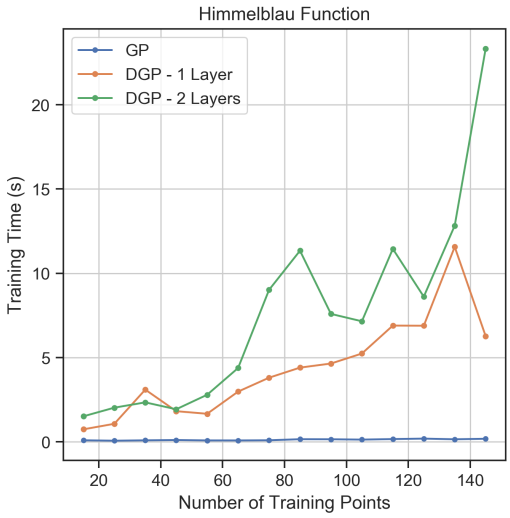
A comparison of GP and DGP is first performed for the Himmelblau test function. More details about this 2-dimensional function are provided in Appendix V. For the Himmelblau test function, as seen in Figures 4a and 4b, it is observed that, as the number of training points increases, the DGP models with 2 layers perform better than the GP

[†]https://github.com/SheffieldML/PyDeepGP

and DGP with 1 layer. The training cost for both variants of DGP models is higher than GPs', which remains relatively constant for this problem even as the number of training points increases considerably. The overall difference in the performance is however, not markedly significant between GP and DGP (especially with a single hidden layer).



(a) Variation of the averaged absolute RMS error with increasing training points

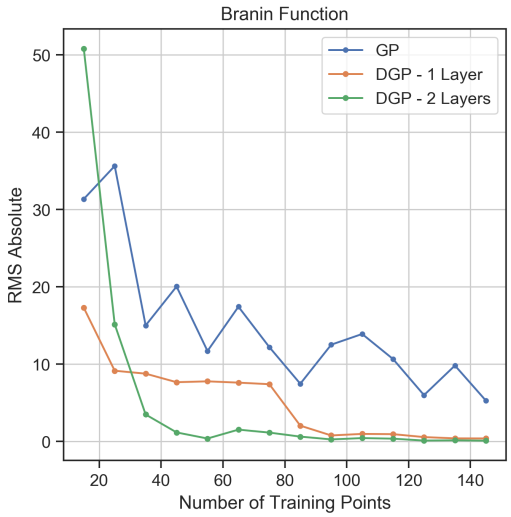


(b) Variation of the averaged training time with increasing training points

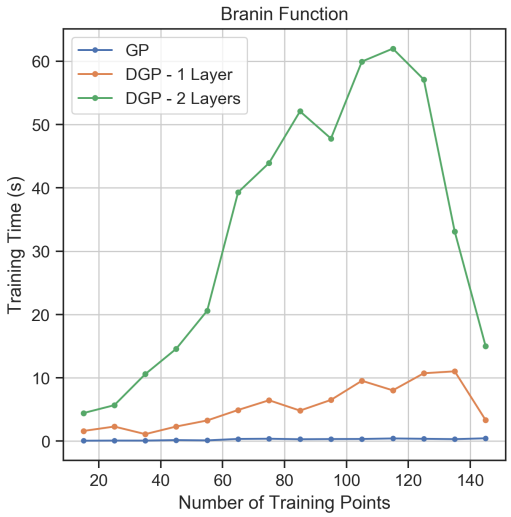
Fig. 4 RMS error and Training time comparisons between GP and DGP for Himmelblau problem

2. Branin Function

In this section, a comparison of GP and DGP is performed for the Branin test function. More details about this 2-dimensional function are provided in Appendix V. For the Branin function, as seen from Figures 5a and 5b, both DGP models exhibit better averaged absolute RMS error than GP models but perform worse than GP in terms of training time with the 2-layer DGP performing considerably worse than the 1-layer DGP. For high number of training points, both variants of DGP perform very well and are almost indistinguishable from each other. The training time however, gets progressively worse as the number of training points increases. This is a general trend that is observed in both the canonical problems explored here as well as in work reported in literature.



(a) Variation of the averaged absolute RMS error with increasing training points



(b) Variation of the averaged training time with increasing training points

Fig. 5 RMS error and Training time comparisons between GP and DGP for Branin problem

As is evident from both these canonical problems, DGP surrogate models hold significant promise in terms of their enhanced performance over traditional GP models, albeit at the cost of additional training time. Due to these reasons, it is important to explore and understand the scalability of DGP models as well as their sensitivity to model hyperparameters in order to use them in the most efficient manner.

B. Scalability Analysis

The next experiment conducted consists of understanding how DGP models scale with higher number of dimensions. The scalability of GPs has been a subject of prior research (Ghosh et al. [16], Eriksson et al. [17], Wilson et al. [18]). However, there has been limited research on scalability of DGPs. The scalability of the method with increased number of dimensions is an important indicator of whether its applications to turbomachinery design problem for example, which have hundreds of design variables. For this purpose, two canonical n-dimensional test functions - Ackley and Trid are utilized. For each function, the number of dimensions can be increased from 2 to n. In each case, for a fixed number of training points (100), DGP models with various hyperparameters (hidden layers, latent units, and inducing points) and an increasing number of dimensions are trained. The measurement of the RMS error is normalized for this particular experiment. This is done because, with increase in the number of dimensions for these functions, the absolute values of the function response are higher. Therefore, it would not be a fair comparison if the absolute RMS error were compared. The average training time is also compared for each case to understand how the training time scales with dimensions.

Figure 6 illustrates the normalized RMS error for the validation points with increasing dimensionality of the Ackley function. As observed from the figure, the normalized error decreases with higher dimensionality for all hyperparameter settings. There is very little difference between the trends and values of the normalized error for the different hyperparameter settings explored in this problem.

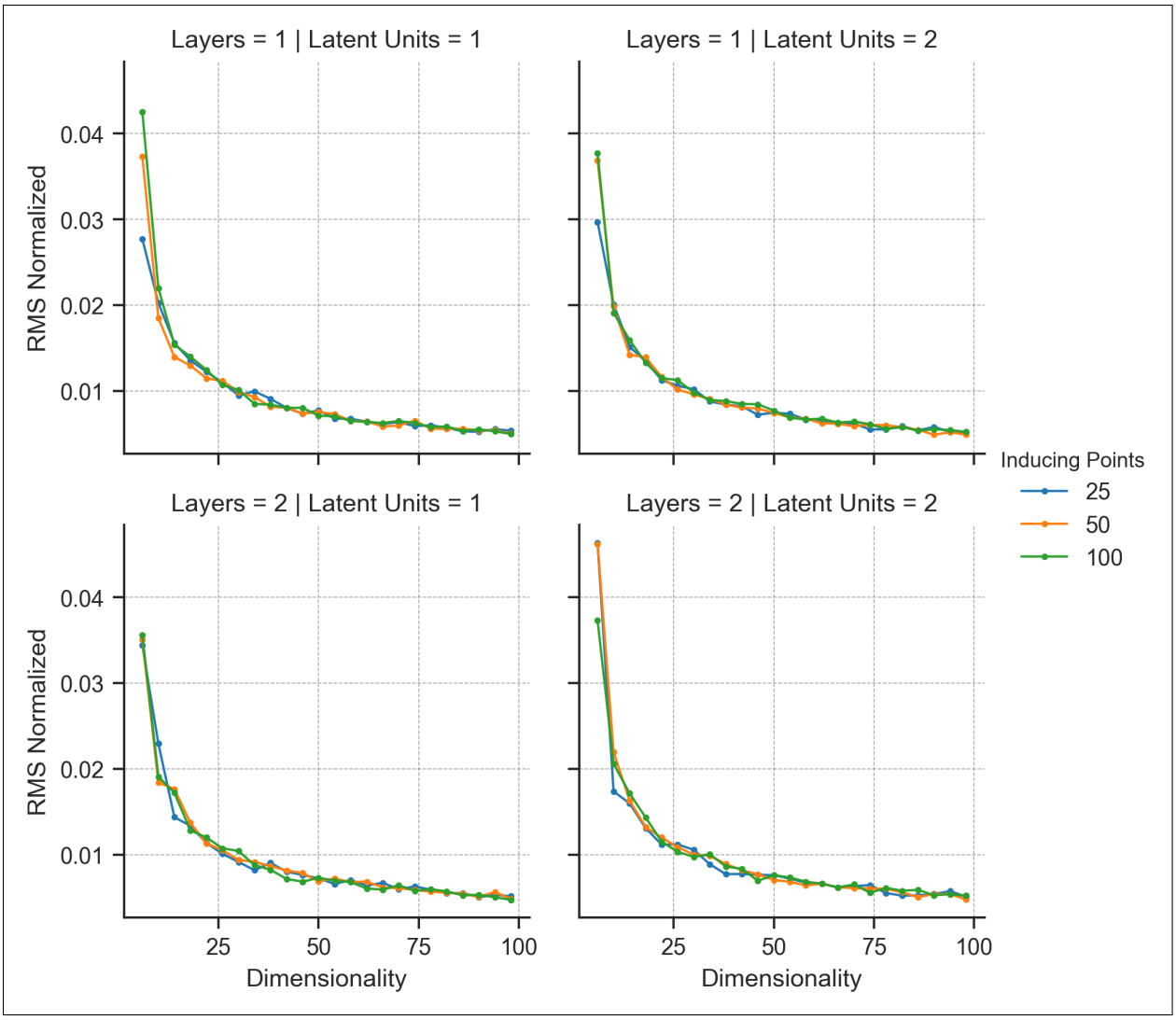


Fig. 6 Root-mean-square validation error for DGP model of Ackley function with increasing dimensionality

Figure 7 shows that the training time remains reasonably constant for a particular level of inducing points as the number of dimensions increases for the Ackley function. Increasing the number of inducing points, however, leads to a noticeable increase in training time. Thus, for a fixed set of hyperparameters and fixed number of training points, as the number of dimensions is increased, the training time remains relatively constant.

Similar to the analysis conducted on the Ackley function, Appendix V contains the scalability results obtained for the Trid canonical function. The same trends are observed in the results for this function indicating that DGP models scale well for higher dimensional problems. Indeed, an interesting trend observed for the two test problems to which DGP has been applied is that higher dimensional problems had a lower normalized error compared to lower dimensional problems. This indicates that DGP is possibly a more suitable model for higher dimensional problems, performing better as the dimensionality increases. For both functions, it was seen that the training time, under this scalability test, is reasonably independent of the hyperparameters (other than the number of inducing points, which has a slight effect on the training time).

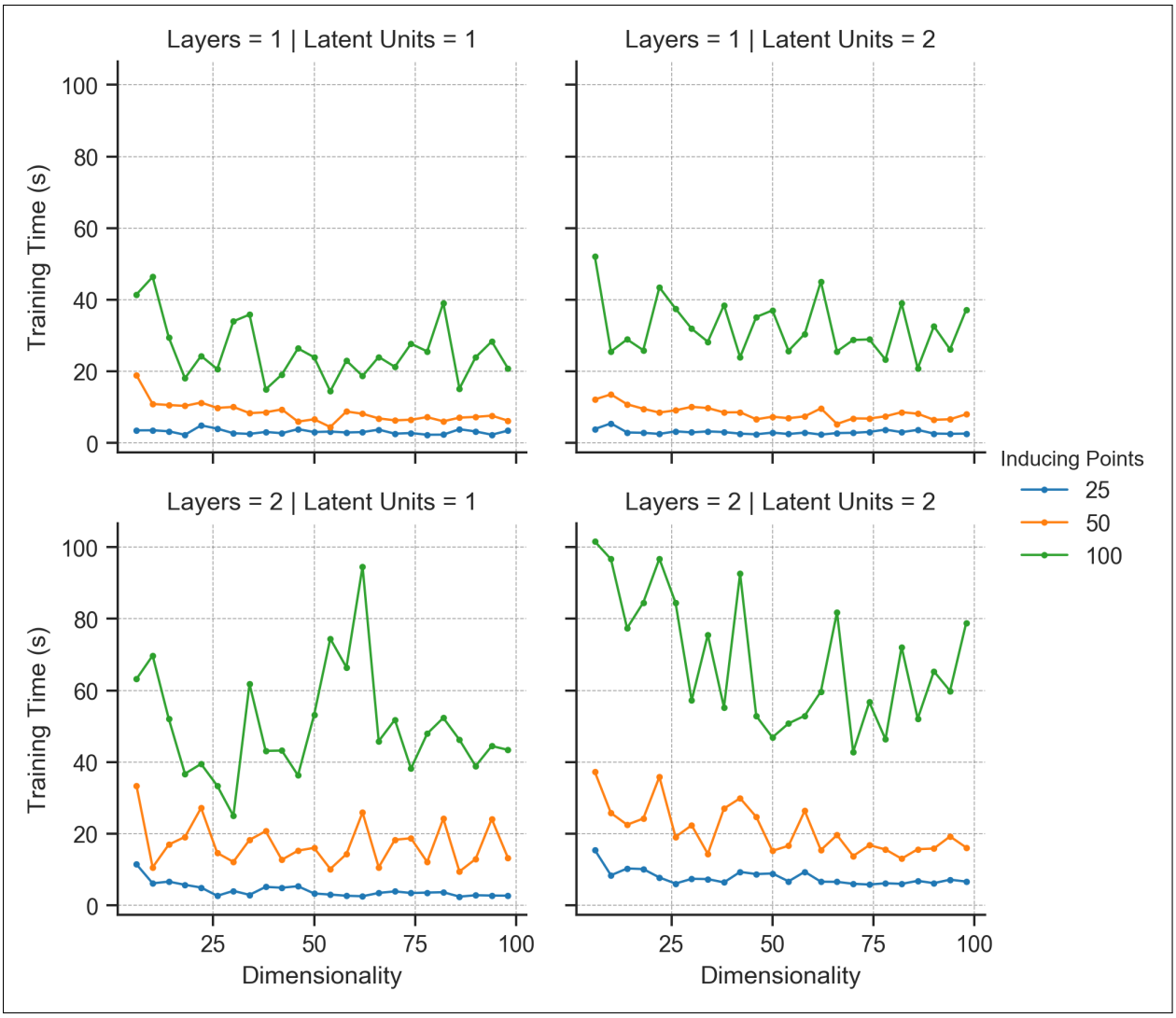


Fig. 7 Scalability of Ackley function with increasing dimensionality

C. Sensitivity to Hyperparameters

While DGP models are demonstrated to be scalable to higher dimensions, they are sensitive to the choice of hyperparameters. For a DGP model, the main hyperparameters affecting the model are the number of hidden layers, the number of units, and the number of inducing points (used in the inference). The combination of these parameters can have an impact on the eventual accuracy and applicability of the method. Therefore, in this section, a methodical approach is undertaken to quantify the sensitivity of the DGP models with respect to these hyperparameters using the canonical functions presented in the section above. In each case, DGP models are trained for the respective function using two different number of training points (50 and 100). For each level of training points, a sweep of the hyperparameter space in the form of a design of experiments (DOE) is conducted along with multiple repetitions at each level. The results obtained from this study are plotted in the figures below. Table 1 outlines the DOE that is executed for the sensitivity analysis. Thus, for each test function, a total of 256 DGP models were trained and validated.

Table 1 Details of design-of-experiments for hyperparameter sensitivity analysis

Parameter	Options / Levels
Number of Hidden Layers	[1, 2]
Number of Latent Units	[1, 2]
Number of Inducing Points	[75, 125, 175, 225]
Number of Training Points	[50, 100]
Number of Repetitions	8

Figure 8 indicates the variability in the validation error for the Himmelblau function using 100 training points. As is evident, the performance of the model varies for different hyperparameter settings. Overall, with a single hidden layer, two latent units generally produce better models than one latent unit for all inducing point settings. This trend is flipped in the two hidden layer models where models with one latent unit outperforms models with two latent units. There is no clear advantage of using more inducing points for this problem within the range of values experimented. It is noted that the best DGP model for 100 training points (which was the model with 2 hidden layers in Figure 4a) is approximately equal to 18 which would be in the lower end of the box plots shown in Figure 8. It is generally observed that two hidden layer models would be better suited for the Himmelblau test function.

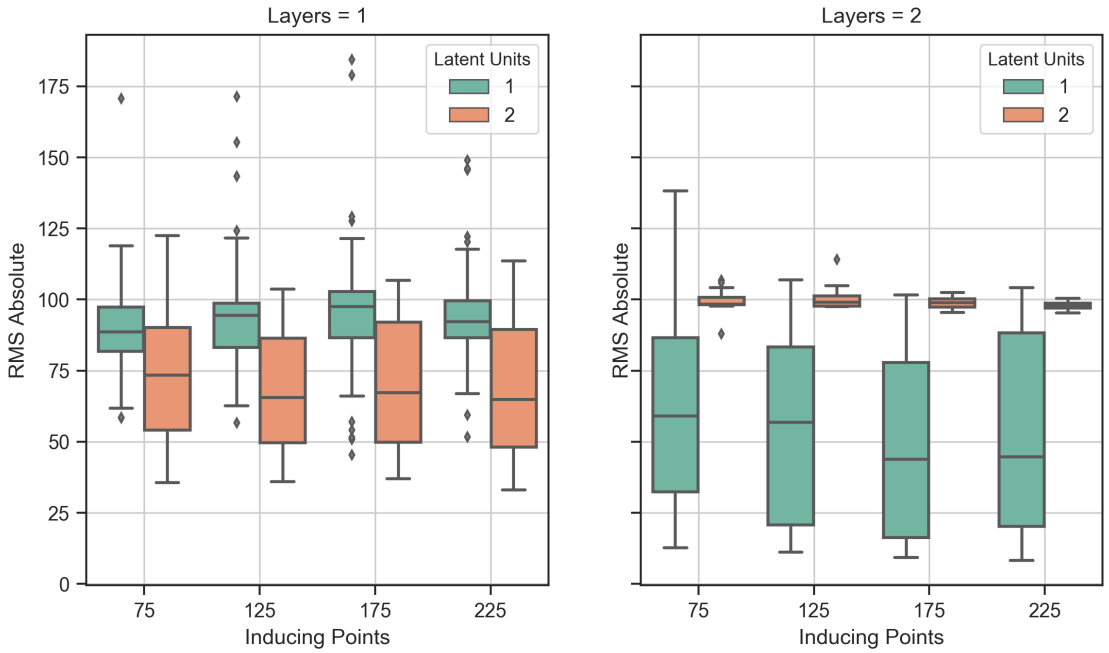


Fig. 8 Sensitivity Analysis for the Himmelblau Test function using 100 training points

For the Branin test function, as observed in Figure 9, the error is slightly higher for all models with a single hidden layer than those with two hidden layers. Models with with single latent units particularly perform poorly for one hidden layer models. Models with higher number of inducing points generally have a tighter distribution in the error for the two-layer variant but the high number of inducing points does not provide any advantage for this problem. It is noted that the best DGP model for 100 training points (which was the model with both 1 and 2 hidden layers in Figure 5a) is approximately equal to 0.02 which would be in the lower end of the box plots shown in Figure 8.

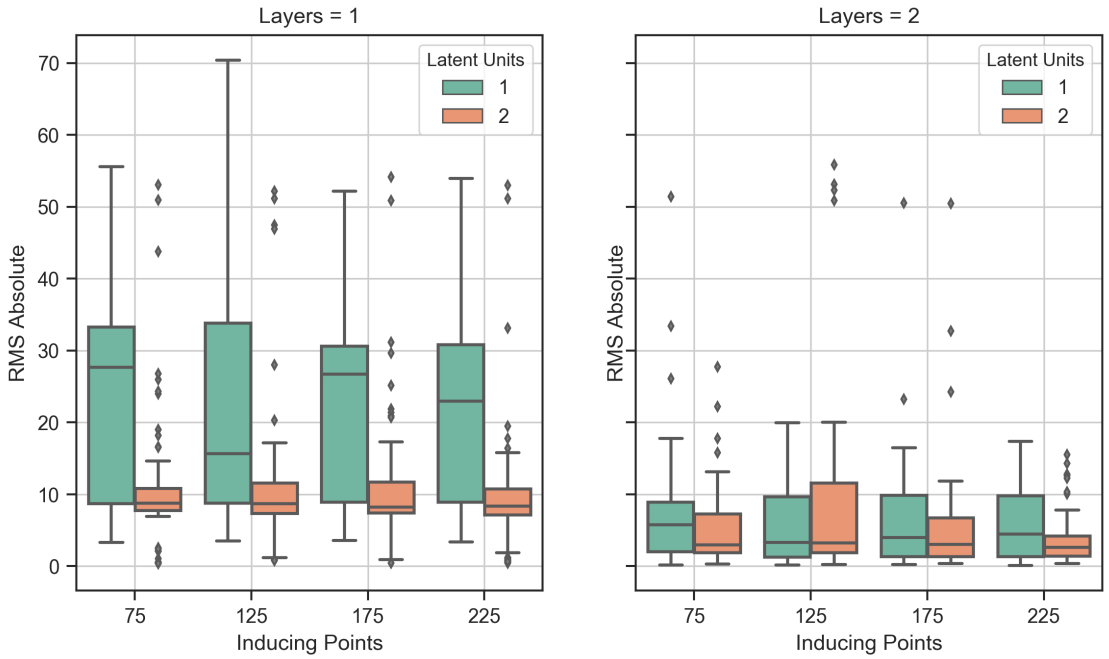


Fig. 9 Sensitivity Analysis for the Branin Test function using 100 training points

An important observation made during these sensitivity experiments is that the number of inducing points chosen seems to have the most impact on whether a good performing model can be obtained. If the number of inducing points is not sufficient, the model tends to under fit the data, no matter the number of training points. If the inducing points are too many, then the training time increases rapidly and the performance does not necessarily improve in the same manner. An observation regarding inducing points is that the number of inducing points required is at least equal to the number of training points and usually slightly higher values worked better. Higher dimensional problems generally performed better with higher number of inducing points. Beyond these, no obvious trends could be observed across the different problems to inform the optimal number of inducing points to be used.

The number of hidden layers and latent units was tested only at two values, 1 and 2, based on previous published literature related to DGP. Among the four combinatorial possibilities of hidden layers and latent units ([1, 1], [1, 2], [2, 1], [2, 2]), both extremes ([1, 1], [2, 2]) usually performed worse. For the lower dimensional canonical problem, models with a single latent unit and single hidden layer had an erratic behavior in the validation error performance. Generally, the trends in performance of the best model for a given number of training points is the same across all four combinations of hidden layers and latent units. Bringing together all the aspects of hyperparameter sensitivity, it can be observed that the performance of DGP is dependent on the hyperparameter settings. However, the nature and behavior of this dependency is not clear. Examining the changes one hyperparameter at a time does not yield any interesting insight.

D. Canonical Engineering Problem

The next set of experiments conducted with DGP models involved canonical engineering problems. In this section, the results obtained from DGP and GP models for the canonical OTL Circuit problem are presented (details about the canonical problem are provided in Appendix V). The function models an output transformerless push-pull circuit and consists of one output and six input dimensions. For this problem, multiple DGP and GP models are trained using a set of training points and the root mean square error on a completely different set of one thousand test points is evaluated. The number of training points is successively increased beginning with 25 training points and the model is trained up to a maximum of 95 training points. For each subset of the training data, GP and DGP models are trained multiple times to account for stochasticity in the training process. In addition, for DGP models, as there are multiple choices available for hyperparameter settings, each DGP model is trained for *eight* hyperparameter settings (full-factorial combinations of $n_{layers} = [1, 2]$, $n_{latent} = [1, 2]$, $n_{inducing} = [75, 150]$). Figures 10 and 11 show respectively, the RMS error and training time required for each of the models at each training point setting.

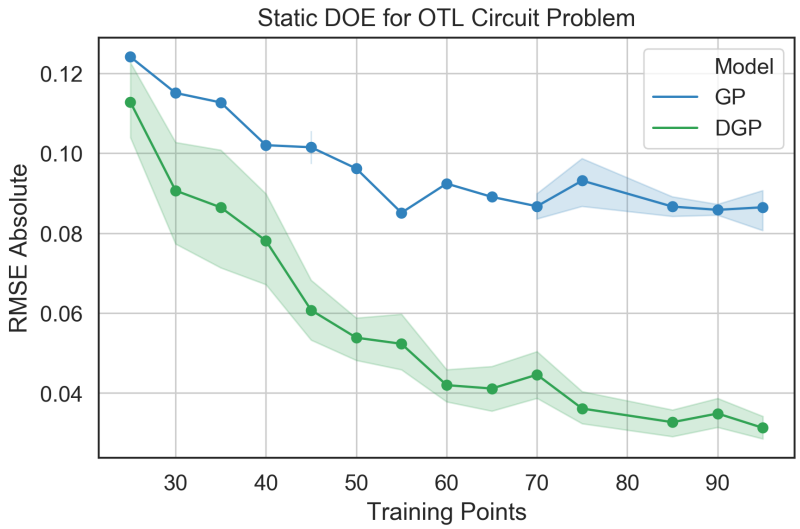


Fig. 10 Average RMS error for each surrogate model with increasing number of training points

Because there are multiple models being trained (repetitions for GP, repetitions and hyperparameter choices for DGP), at each training point setting, the average error and training time are computed. The shaded region represents the spread of values obtained at the same training point setting for different models. It is apparent that there is a larger spread among the values of DGP models as the different hyperparameter settings can significantly affect the model quality. For each training point setting in this 6-dimensional problem, DGP models perform better than GP models throughout. The overall trend observed for both models is that the performance improves as the number of training points increases, although this is more pronounced for DGP than GP.

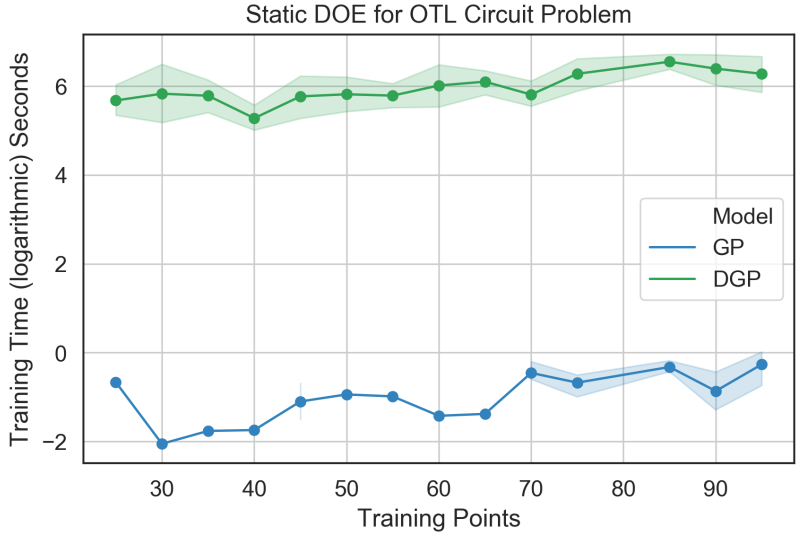


Fig. 11 Average training time required for each surrogate model on a natural log scale with increasing number of training points.

In terms of training time, it is observed that DGP models have a significantly worse computational performance than GP models for the same number of training points. The average training time does not increase noticeably for DGP for higher number of training points whereas it increases by an order of magnitude for GP models. These results indicate that, for the canonical engineering problem, the cost of DGP models is higher throughout than that of GP models. Table 2 shows the results of the best model obtained at each training point setting for each of the two types of

surrogate models (DGP and GP) along with the percentage difference between the two (measured as $\left(\frac{GP-DGP}{GP}\right) \times 100$).

Table 2 Summary of results on OTL circuit problem with static DOE

OTL Circuit Problem							
Training Points	RMSE GP	RMSE DGP	Difference	Training Time GP (s)	Training Time DGP (s)	Difference	Speed-up DGP
25	0.124	0.094	24%	0.514	159.2	-30,872%	3×10^{-3}
30	0.115	0.056	51%	0.341	80.06	-23,378%	4×10^{-3}
35	0.112	0.052	53%	0.248	147.7	-54,456%	1×10^{-3}
40	0.102	0.056	45%	0.286	114.1	-39,795%	2×10^{-3}
45	0.097	0.046	52%	0.506	77.88	-15,291%	6×10^{-3}
50	0.096	0.040	58%	0.391	160.3	-40,897 %	2×10^{-3}
55	0.085	0.044	48%	0.443	163.5	-36,807 %	2×10^{-3}
60	0.092	0.033	64%	0.423	173.6	-40,940 %	2×10^{-3}
65	0.089	0.025	72%	0.278	216.6	-77,814 %	1×10^{-3}
70	0.083	0.032	61%	0.817	231.7	-28,260 %	3×10^{-3}
75	0.086	0.031	64%	0.592	210.6	-35,474 %	2×10^{-3}
80	0.082	0.032	61%	0.982	180.3	-18,260 %	5×10^{-3}
85	0.083	0.023	72%	0.887	474.7	-53,417 %	1×10^{-3}
90	0.084	0.027	68%	0.645	310.7	-48,071 %	2×10^{-3}
95	0.088	0.025	72%	1.016	283.9	-27,843 %	3×10^{-3}

As is evident from the table, the best performing DGP model is better than the best performing GP model but is computationally much more expensive than the corresponding GP model. Therefore, this presents an interesting trade-off that will eventually determine which model ought to be used in a particular setting.

One of the drawbacks of using a static DOE to train the surrogate models is that information about the training points already available is not intelligently utilized in training newer models. Because of this, multiple static DOEs need to be computed and the model trained multiple times in order to get a model that performs well. This process is computationally expensive and does not necessarily guarantee a good model every time. One of the ways of overcoming some of these limitations is using adaptive sampling for identifying the next training point to be sampled.

E. Adaptive Sampling

While static design of experiments are well suited for a given set of pre-evaluated designs, adaptive strategies provide a principled feed-back procedure to recommend new experiments serially as new information about the underlying function's landscape becomes available. In this section, the value of DGPs in comparison to GPs is assessed in closed-loop or adaptive scenarios. However, since closed-loop strategies are inherently serial in nature, a major drawback is the inability of parallelization for evaluating the function to be emulated.

Recommendations for new designs to be evaluated are obtained by optimizing some criterion (defined over the design space) that maximizes the information that can be gained about the function. Several criteria have been proposed and assessed for use in closed-loop settings [19–21]. In this paper, the *maximum variance design* criterion is maximized over the design space in order to guide the search. The criterion capitalizes on the idea that the best location to sample is the one with the largest uncertainty in the prediction. For GPs and DGPs, the *maximum variance design* results by maximizing the posterior variance (conditioned on the observed data). Although a closed-form expression is not available for DGPs (due to the composition of multiple GPs), the approximate variational inference allows evaluation of the posterior variance. The expression for posterior variance for GPs is given by:

$$s^2(\mathbf{x}) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}_*^T M^{-1} \mathbf{k}_*, \quad \text{where} \quad M = K + \sigma_n^2 I$$

In the expression above, $k(\cdot,\cdot)$ is the covariance function, $\mathbf{x}_*$ is the test point where the posterior variance must be evaluated, and K is the matrix containing pairwise covariances computed for the points observed so far. The performance of adaptive sampling using GP and DGP is demonstrated on the OTL circuit canonical engineering problem and the aerodynamic shape problem. Beginning with a GP/DGP trained on a static latin hypercube design with a pre-specified number of points, the procedure proceeds by maximizing the posterior variance to yield the next design point to be evaluated and added to the training set. The models are then updated with the new observed data point by warm-starting the training algorithm with the parameter values of the current model. This process is repeated until a specified number of training points are added to both the GP and the DGP models. In order to avoid getting trapped in local optima, this work employs the meta-heuristic particle swarm optimization (PSO) algorithm to optimize over the posterior variance.

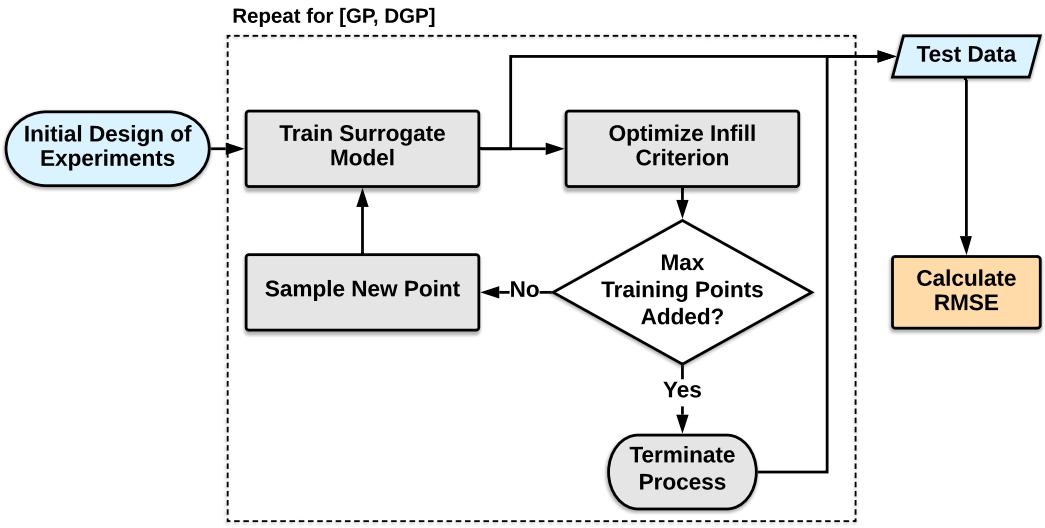


Fig. 12 Adaptive Sampling Methodology

The adaptive sampling strategy highlighted in Figure 12 is first applied to the canonical OTL circuit engineering problem. For this particular problem, an initial candidate model at a particular set of training points is chosen for each of the two surrogate models. In order to have a fair comparison across models, the model structure (hyperparameters, kernel, etc.) is fixed at the same configuration and new points are added adaptively as outlined earlier and the RMS error on the same test set as the previous experiment with static DOE is calculated. Figure 13 shows the results obtained from application of adaptive sampling on the OTL circuit problem.

In this experiment, a model obtained from the static DOE using 30 initial training points is chosen as the start and 30 more training points are adaptively added while retraining the model after every addition. The progression of the RMS error with each added training point is shown in Figure 13 using the solid green (DGP) and blue (GP) lines. The dotted green (DGP) and blue (GP) lines represent the average RMSE for the initial static models with 30 points. Similarly, the dashed green line (DGP) and dashed blue line (GP) represent the average RMSE for the static models with 60 training points. These dotted and dashed lines represent average errors with no adaptive sampling (static DOEs). As is evident from Figure 13, the performance of the GP and DGP models steadily improves with adaptively added training points to the point where both models perform better than the static models with the same number of training points. It is noted however, that for the static models, the performance represented by dashed lines is obtained following a search across multiple hyperparameters and repetitions, which requires significant computational resources, whereas for the adaptive sampling, the same model is retrained with the added points and therefore is computationally much more efficient.

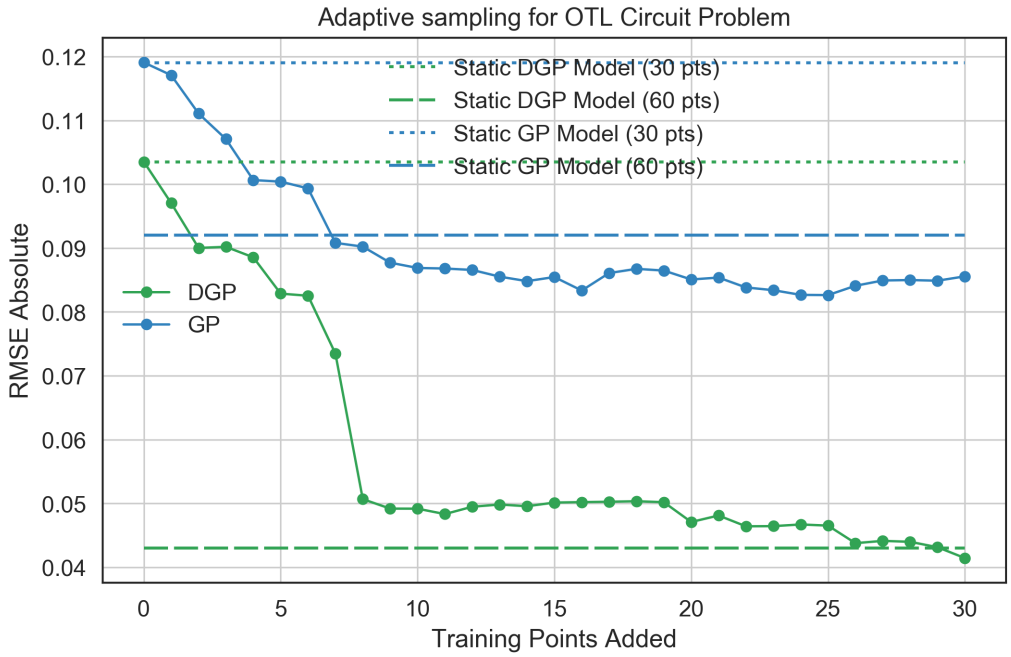


Fig. 13 Adaptive sampling for DGP and GP models using 30 initial training points

F. Aerodynamic Shape Problem

Using the knowledge from the canonical test problems and engineering problem, DGP models are now applied to the airfoil aerodynamics problem described earlier in section III. For this problem, parameterized airfoils are constructed using 42-dimensional design vectors and the flow properties are evaluated using XFoil. For the purpose of demonstration in this paper, the non-dimensional lift coefficient (C_L) and non-dimensional drag coefficient (C_D) of the airfoil are chosen as the output quantities of interest. Both GP and DGP models are successively trained using an increasing number of airfoil shapes in the training set and the error is evaluated on an independent test set of a thousand airfoil shape outputs. Similar to the OTL circuit problem, each DGP model is trained for *eight* hyperparameter settings (full-factorial combinations of n_{layers} = [1, 2], n_{latent} = [1, 2], $n_{inducing}$ = [75, 150]).

1. Static DOE for Airfoil Lift Coefficient Problem

Figures 14 and 15 show respectively, the RMS error and training time required for each of the models at each training point setting for the airfoil lift coefficient problem. It is observed that, the DGP models afford better accuracy but poorer computational performance. It is noted that for both GP and DGP there is some spread observed in the errors at each training point setting. The performance generally improves with increased number of training points.

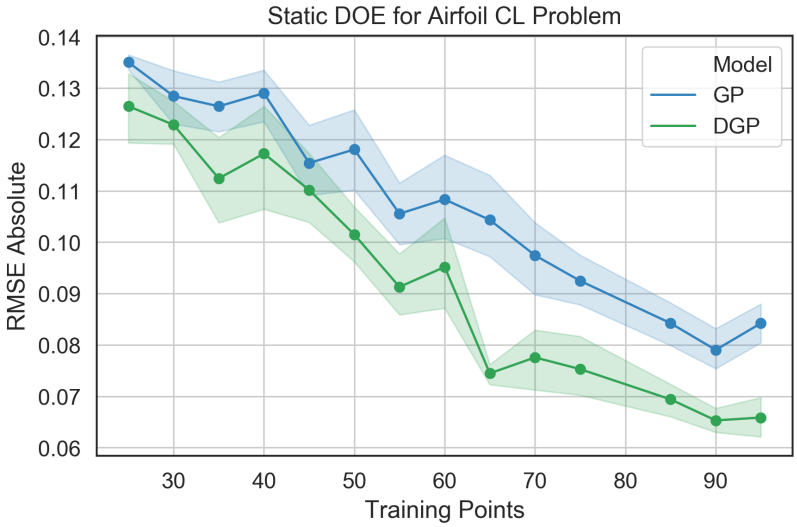


Fig. 14 Average RMS error for each surrogate model with increasing number of training points for the airfoil lift coefficient problem

The computational cost of the GP and DGP models for this 42-dimensional problem is similar to that of the lower dimensional OTL circuit problem and remains reasonably constant as the number of training points increase. It can also be seen that GP models perform significantly better than DGP models in terms of computational cost.

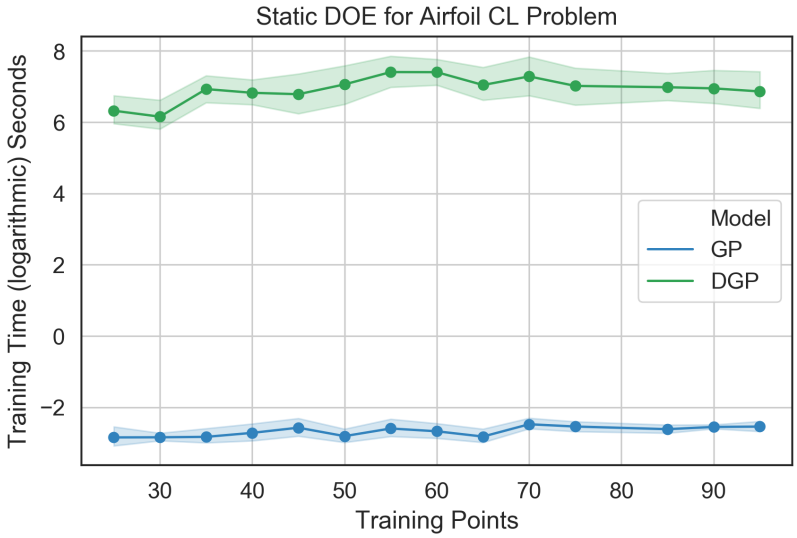


Fig. 15 Average training time for each surrogate model with increasing number of training points for the airfoil lift coefficient problem

Table 3 shows the results of the best model obtained at each training point setting for each of the two types of surrogate models (DGP and GP) along with the percentage difference between the two (measured as $\left(\frac{GP-DGP}{GP}\right) \times 100$).

Table 3 Summary of results on airfoil lift coefficient problem with static DOE

Airfoil CL Problem							
Training Points	RMSE GP	RMSE DGP	Difference	Training Time GP (s)	Training Time DGP (s)	Difference	Speed-up DGP
25	0.133	0.103	23%	0.132	274.9	-208,158%	4×10^{-4}
30	0.117	0.113	3%	0.064	183.4	-285,838 %	3×10^{-4}
35	0.113	0.091	19%	0.129	479.7	-371,218%	2×10^{-4}
40	0.112	0.094	16%	0.136	538.8	- 395,488%	2×10^{-4}
45	0.104	0.102	2%	0.104	320.8	- 307,592%	3×10^{-4}
50	0.099	0.087	12%	0.102	348.3	- 341,076%	2×10^{-4}
55	0.092	0.051	45%	0.146	685.5	- 469,078%	2×10^{-4}
60	0.093	0.082	12%	0.111	743.0	- 669,269%	1×10^{-4}
65	0.088	0.068	23%	0.112	458.1	- 408,829%	2×10^{-4}
70	0.081	0.065	20%	0.147	462.1	- 314,186%	3×10^{-4}
75	0.082	0.066	20%	0.117	486.2	- 415,285%	2×10^{-4}
80	0.078	0.061	22%	0.072	514.9	- 713,789%	1×10^{-4}
85	0.072	0.059	18%	0.098	532.7	- 542,757%	1×10^{-4}
90	0.071	0.059	17%	0.091	564.7	- 619,680%	1×10^{-4}
95	0.077	0.057	26%	0.102	590.6	- 578,331%	1×10^{-4}

As seen from the table, there is a marked improvement in the overall accuracy and a marked decline in the computational performance when using DGP over GP for higher dimensional problems.

2. Static DOE for Airfoil Drag Coefficient Problem

Finally, the static DOE is tested on the other output quantity of interest in the aerodynamic test case: the non-dimensional drag coefficient. Figures 16 and 17 show respectively, the RMS error and training time required for each of the models at each training point setting for the airfoil drag coefficient problem. The results and trends obtained mirror those from the lift coefficient and OTL circuit problems.

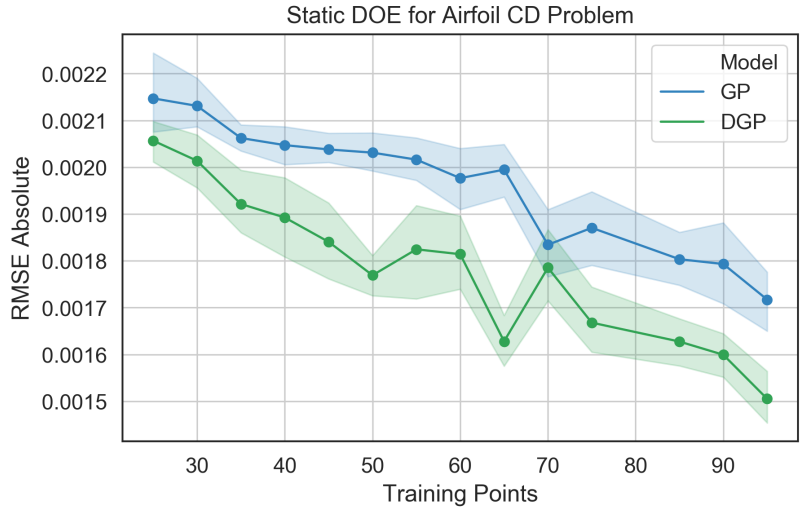


Fig. 16 Average RMS error for each surrogate model with increasing number of training points for the airfoil drag coefficient problem

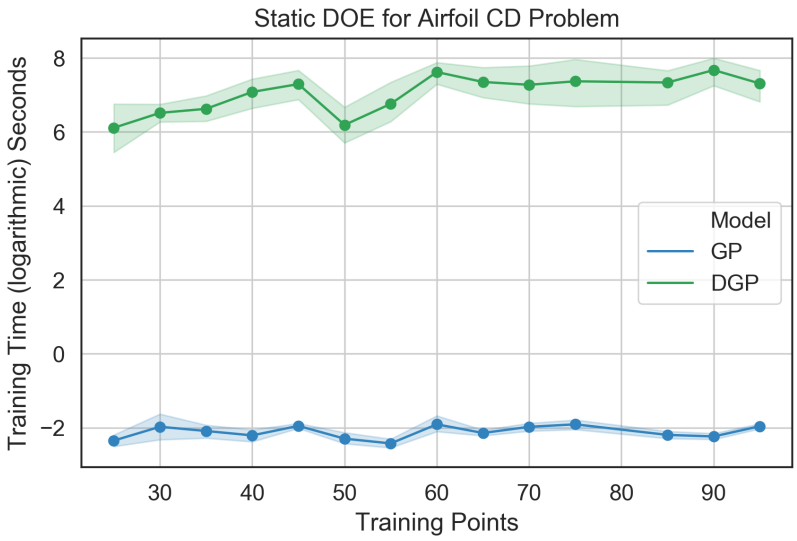


Fig. 17 Average training time for each surrogate model with increasing number of training points for the airfoil drag coefficient problem

Table 4 shows the results of the best model obtained at each training point setting for each of the two types of surrogate models (DGP and GP) along with the percentage difference between the two (measured as $\left(\frac{GP-DGP}{GP}\right) \times 100$).

Table 4 Summary of results on airfoil drag coefficient problem with static DOE

Airfoil CD Problem							
Training Points	RMSE GP	RMSE DGP	Difference	Training Time GP (s)	Training Time DGP (s)	Difference	Speed-up DGP
25	0.00198	0.00194	2%	0.133	92.15	-69,073%	1×10^{-3}
30	0.00204	0.00190	7%	0.211	548.2	-259,616%	3×10^{-4}
35	0.00202	0.00176	13%	0.162	387.33	-238,789%	4×10^{-4}
40	0.00192	0.00169	12%	0.133	468.8	-351,780%	2×10^{-4}
45	0.00199	0.00167	16%	0.149	697.5	-467,685%	2×10^{-4}
50	0.00195	0.00168	14%	0.161	359.3	-222,881%	4×10^{-4}
55	0.00193	0.00160	17%	0.121	387.0	-319,735%	3×10^{-4}
60	0.00188	0.00173	8%	0.303	739.9	-243,794%	4×10^{-4}
65	0.00185	0.00150	19%	0.141	603.9	-427,560%	2×10^{-4}
70	0.00170	0.00164	4%	0.167	455.6	-272,355 %	3×10^{-4}
75	0.00176	0.00155	12%	0.205	251.8	-122,339 %	8×10^{-4}
80	0.00172	0.00154	10%	0.273	714.7	-261,438 %	3×10^{-4}
85	0.00168	0.00149	11%	0.141	182.9	-129,616 %	7×10^{-4}
90	0.00162	0.00146	10%	0.125	558.2	-446,460 %	2×10^{-4}
95	0.00153	0.00142	7%	0.160	766.3	-478,838 %	2×10^{-4}

As seen from the table, there is a marked improvement in the overall accuracy and a marked decline in the computational performance when using DGP over GP for higher dimensional problems. It is worth noting that the computational time for both GP and DGP is almost the same for 42 dimensional problem as it is for the corresponding GP and DGP model used in the 6 dimensional problem. This observation is consistent with the results of the scalability

study presented is section IV.B. Dealing with functions with a large number of variables may demand a large number of evaluations on the original function to build a reliable global metamodel. Hence, the computational time associated with the metamodel building can be prohibitive, especially if there is a high computational cost on the function evaluation. This is where surrogate models that work well on higher dimensional problems such as DGP are valuable as they can provide reliable models with small data. With this understanding, it is important to explore the benefits of adaptive sampling demonstrated on the OTL circuit problem for the airfoil aerodynamic shape problem. In the next two subsections, the results obtained from adaptive sampling for the airfoil lift and drag coefficient problems are presented.

3. Adaptive Sampling for Airfoil Lift Coefficient Problem

The adaptive sampling strategy highlighted in Figure 12 is now applied to the airfoil lift coefficient problem. Similar to the OTL circuit problem, an initial candidate model at a particular set of training points (40 points) is chosen for each of the two surrogate models. In order to have a fair comparison across models, the model structure (hyperparameters,kernel, etc.) is fixed at the same configuration and new points are added adaptively. The RMS error on the same test set as the experiment using the static DOE is then calculated. Figure 18 shows the results obtained from applying adaptive sampling on the airfoil lift coefficient problem. A total of 30 adaptive samples are added to the original set of 40 points while retraining the model at each step and calculating the RMS error on the test set.

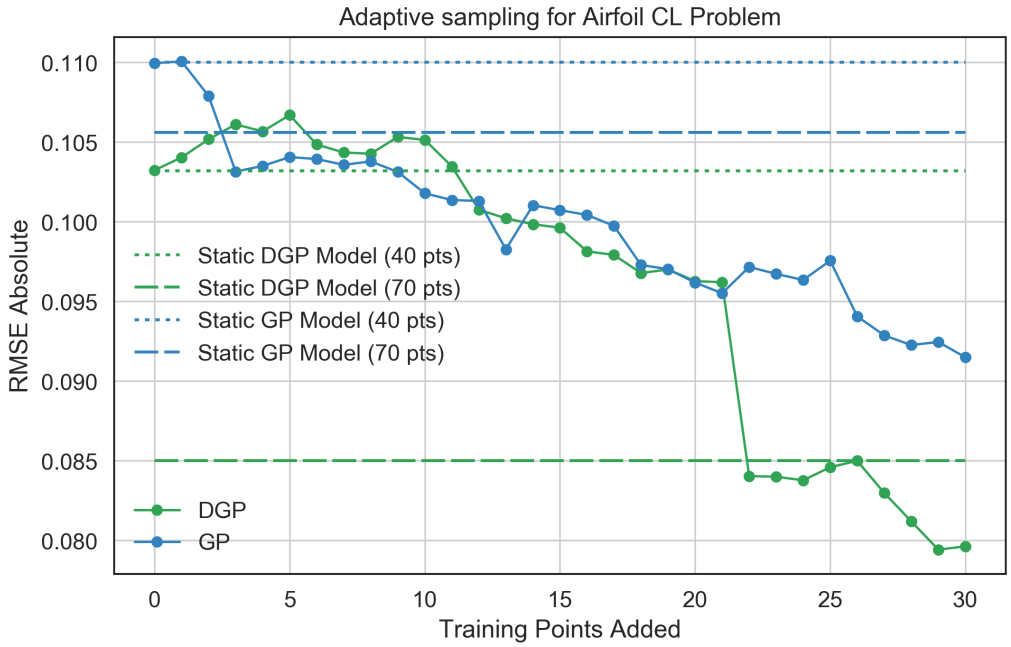


Fig. 18 Adaptive Sampling for Lift Coefficient Problem

As is evident from Figure 18, the performance of the GP and DGP models steadily improves when adaptively adding training points and both models performing better than the static models with the same total number of training points (70 points). It is noted however, that for the static models, the performance of the dashed line is obtained after a search across multiple hyperparameters and repetitions which requires significant computational resources whereas for the adaptive sampling the same model is retrained with the added points and therefore is computationally much more efficient. The adaptive sampling DGP models with 70 points are $\sim 6\%$ better than the static DGP model for 70 points despite using a fixed hyperparameter setting, whereas the adaptive sampling GP models are $\sim 12\%$ better than the static GP models for 70 training points. Overall, the computational efficiency of using adaptive sampling over a static DOE coupled with the superior performance of DGP over GP make it a compelling combination to be used for high-dimensional problems.

4. Adaptive Sampling for Airfoil Drag Coefficient Problem

The final set of results shown in this work are for the application of the adaptive sampling scheme to the other aerodynamic quantity of interest - the non-dimensional drag coefficient. In this case, an initial candidate model at a particular set of training points (30 points) is again chosen for each of the two surrogate models. In order to have a fair comparison across models, the model structure (hyperparameters,kernel, etc.) is fixed at the same configuration and new points are added adaptively. The RMS error is then calculated on the same test set as the experiment with static DOE is calculated. Figure 19 shows the results obtained from applying the adaptive sampling on the airfoil drag coefficient problem. A total of 30 adaptive samples are added to the original set of 30 points while retraining the model at each step and calculating the RMS error on the test set.

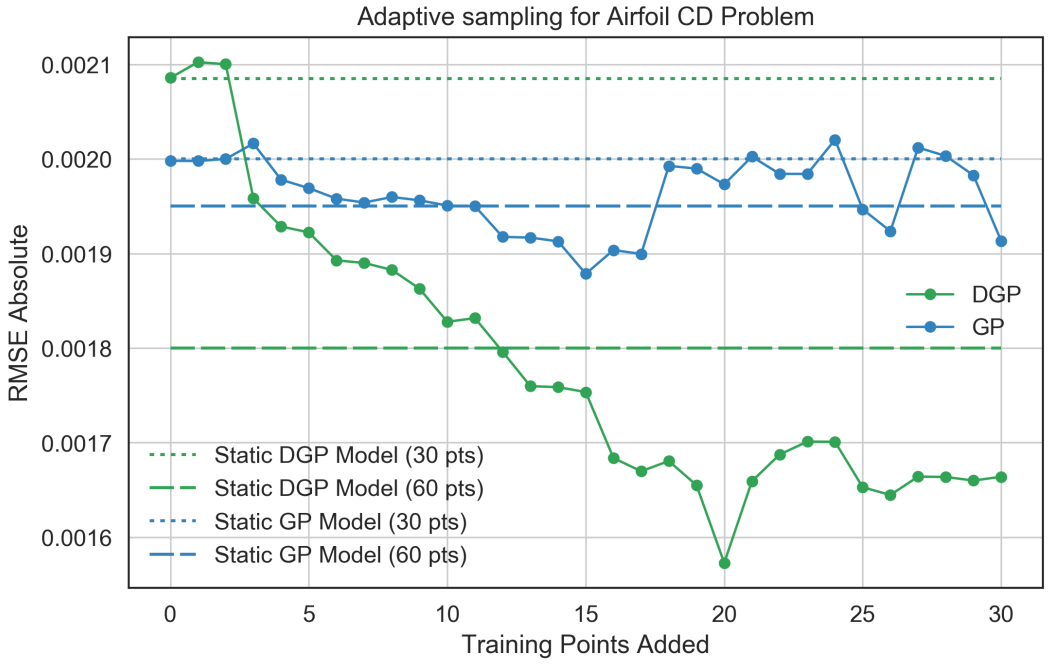


Fig. 19 Adaptive Sampling for Drag Coefficient Problem

The accuracy of the GP and DGP models improves with additional training points for the airfoil drag coefficient problem. However, for this problem, it is noted that both the static and adaptive sampling based GP models do not improve much over the range of training points tested. The final adaptive GP model is only $\sim 2.5\%$ better than the final static GP model with 60 points. For DGP model on the other hand, the static model improves by $\sim 14\%$ from 30 to 60 training points whereas the adaptive sampling DPG model improves by $\sim 21\%$ from the initial 30 training points to the final model with 30 adaptively added points. Additionally, the adaptive sampling DGP model surpasses the performance of the static GP model with 60 training points after adaptively adding just 3 training points. It also surpasses the performance of the static DGP model with 60 training points after adaptively adding 11 training points. This indicates that, for the more complex drag coefficient prediction surrogate model, the DGP model with adaptive sampling provides a great improvement in the accuracy with relatively a low number of added training points.

V. Conclusion and Future Work

This paper has demonstrated the potential benefits, value, and limitations of DGP as a surrogate model for problems characterized by high input dimensionality, small data, and complex behavior such as aerodynamic flows. It did so by demonstrating the systematic application of DGP and comparing it to a similar surrogate model (traditional GP) for a series of problems of increasing complexity. Throughout the problems, the improved accuracy of DGP over GP, especially for high dimensional problems was observed. A scalability study was conducted for DGP which showed that the model tends to scale well with increasing dimensions and performs reasonably well for small data sets. A thorough sensitivity study of the various hyperparameters of DGP models was also conducted which showed that the

predictive accuracy of DGP models and their computational cost of training is influenced by the models' hyperparameter settings. Finally, the value of adaptive sampling for both DGP and GP models using canonical and practical problems was assessed. Based on the results presented in this work, DGP models offer increased accuracy over traditional GPs for all the problems explored at the cost of additional computational time. Therefore, DGPs are particularly suitable for problems with small data sets such as aerodynamic surrogate models where sufficient computational resources are available to train and validate the models but there is a significant incremental cost to get additional data points (either through high-fidelity simulation like computational fluid dynamics or actual experiments). DGPs used on small data with adaptive sampling were shown to have significant promise for high-dimensional problems.

In future work, the benchmarking will further be extended to include other state-of-the-art surrogate models such as DNNs. The models will also be tested with higher fidelity aerodynamic simulation outputs. Once DGPs have been successfully applied to scalar aerodynamic quantities of interest, they could also be used for approximating variation of latent space coordinate in the case of reduced order models such as those in parallel and previous work [22, 23]. Another important avenue of future work that is being investigated is inclusion of hyperparameter tuning in the adaptive sampling scheme or separate hyperparameter optimization for the surrogate model to improve performance.

Acknowledgements

The authors would like to acknowledge the support of Siemens Corporate Technology for the work performed in this paper. In particular, we would like to thank Dr. Sanjeev Srivastava and Dr. Wei Xia from Siemens for their valuable technical feedback and support. This material is based upon work partially supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357.

References

[1] Damianou, A., and Lawrence, N., "Deep Gaussian Processes," *Artificial Intelligence and Statistics*, 2013, pp. 207–215. URL http://proceedings.mlr.press/v31/damianou13a.pdf.

[2] Damianou, A., "Deep Gaussian Processes and Variational Propagation of Uncertainty," Ph.d. thesis, University of Sheffield, 2015. doi:10.13140/RG.2.1.1458.8885.

[3] Vafa, K., and Rush, A., "Training and Inference for Deep Gaussian Processes," Ph.D. thesis, Harvard University, 2016.

[4] Zhang, Y., Ghosh, S., Asher, I., Ling, Y., and Wang, L., "Learning Uncertainty using Clustering and Local Gaussian Process Regression," *AIAA Scitech 2019 Forum*, 2019. doi:10.2514/6.2019-1730.

[5] Bui, T. D., Hernández-Lobato, D., Li, Y., Hernández-Lobato, J. M., and Turner, R. E., "Deep Gaussian Processes for Regression using Approximate Expectation Propagation," *Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016.*, Vol. 48, New York, NY, USA, 2016, pp. 1472–1481. doi:10.1162/NECO_a_00104.

[6] Salimbeni, H., and Deisenroth, M., "Doubly Stochastic Variational Inference for Deep Gaussian Processes," *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, 2017, pp. 4588–4599. URL http://arxiv.org/abs/1705.08933.

[7] Hebbal, A., Brevault, L., Balesdent, M., Talbi, E.-G., and Melab, N., "Multi-objective optimization using Deep Gaussian Processes: Application to Aerospace Vehicle Design," *AIAA SciTech Forum*, San Diego, CA, 2019, pp. 1–19. doi:10.2514/6.2019-1973.

[8] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D., "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, Vol. 112, No. 518, 2017, pp. 859–877.

[9] Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I., "An introduction to MCMC for machine learning," *Machine learning*, Vol. 50, No. 1-2, 2003, pp. 5–43.

[10] Drela, M., "XFOIL: An Analysis and Ddesign System for Low Reynolds Number Airfoils," *Low Reynolds number aerodynamics*, Springer, 1989, pp. 1–12.

[11] Ghosh, S., Ran, H., and Mavris, D. N., "A Generic Airfoil Design Method Based on a Naturally Bounded PARSEC Approach," *32nd AIAA Applied Aerodynamics Conference*, 2014, p. 2010. doi:10.2514/6.2014-2010.

[12] Kulfan, B., and Bussoletti, J., "Fundamental Parameteric Geometry Representations for Aircraft Component Shapes," *11th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2006, p. 6948. doi:10.2514/1.29958.

[13] Kulfan, B. M., "Universal parametric geometry representation method," *Journal of Aircraft*, Vol. 45, No. 1, 2008, pp. 142–158.

[14] Sung, W. J., "A neural network construction method for surrogate modeling of physics-based analysis," Ph.D. thesis, Georgia Institute of Technology, 2012. URL http://hdl.handle.net/1853/43721.

[15] Zhang, Y., Sung, W. J., and Mavris, D. N., "Application of convolutional neural network to predict airfoil lift coefficient," *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018. doi:10.2514/6.2018-1903.

[16] Ghosh, S., Asher, I., Kristensen, J., Ling, Y., Ryan, K., and Wang, L., "Bayesian Multi-Source Modeling with Legacy Data," *2018 AIAA Non-Deterministic Approaches Conference*, 2018. doi:10.2514/6.2018-1663.

[17] Eriksson, D., Dong, K., Lee, E., Bindel, D., and Wilson, A. G., "Scaling Gaussian process regression with derivatives," *Advances in Neural Information Processing Systems*, 2018, pp. 6867–6877. URL http://papers.nips.cc/paper/7919-scaling-gaussian-process-regression-with-derivatives.pdf.

[18] Wilson, A. G., Dann, C., and Nickisch, H., "Thoughts on massively scalable Gaussian processes," *arXiv preprint arXiv:1511.01870*, 2015. URL https://arxiv.org/pdf/1511.01870.pdf.

[19] Garbo, A., and German, B., "Comparison of adaptive design space exploration methods applied to S-duct CFD simulation," *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2016. doi:10.2514/6.2016-0416.

[20] Ghosh, S., Kristensen, J., Zhang, Y., Subber, W., and Wang, L., "A Strategy for Adaptive Sampling of Multi-Fidelity Gaussian Processes to Reduce Predictive Uncertainty," *ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers Digital Collection, 2019. doi:10.1115/DETC2019-98418.

[21] Rajaram, D., and Pant, R., "An improved methodology for airfoil shape optimization using surrogate based design optimization," *Engineering Optimization*, Vol. IV, 2015, p. 147–152.

[22] Rajaram, D., Puranik, T. G., Perron, C., and Mavris, D. N., "Non-Intrusive Parametric Reduced Order Modeling using Randomized Algorithms," *AIAA Scitech Forum*, 2020.

[23] Renganathan, S. A., Liu, Y., and Mavris, D. N., "Koopman-Based Approach to Nonintrusive Projection-Based Reduced-Order Modeling with Black-Box High-Fidelity Models," *AIAA Journal*, Vol. 56, No. 10, 2018, pp. 4087–4111. doi:10.2514/1.J056812.

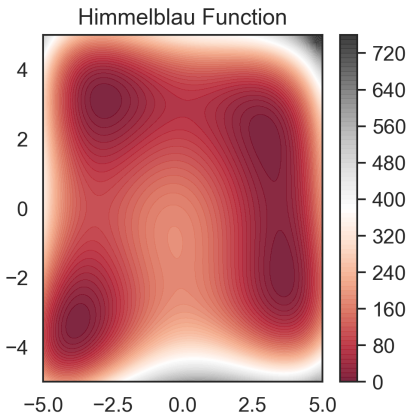
Appendix: Canonical Problems

In this section, additional details about the canonical functions used in this work are provided.

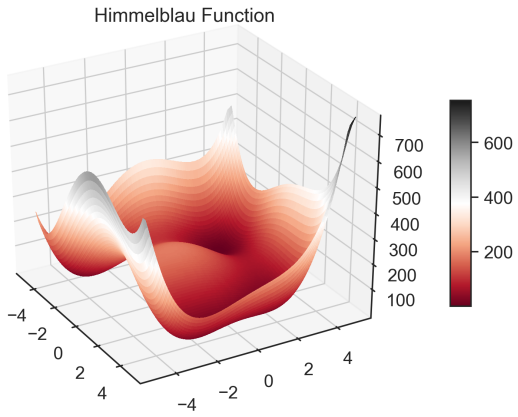
Himmelblau Function

The Himmelblau function is a commonly used analytical function of two dimensions that is continuous and non-convex. The function is usually defined on an input domain of [-5, 5] for both input dimensions. The function has four local minima located at [3, 2], [-2.8051, 3.2832], [-3.7793, -3.2832], and [3.5485, -1.8481]. The function contours and three dimensional surface plot are shown in Figures 20a and 20b respectively. The analytical form of this function is given by the following equation:

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{6}$$



(a) Contours of the Himmelblau function in its domain



(b) Surface plot of the Himmelblau function in its domain

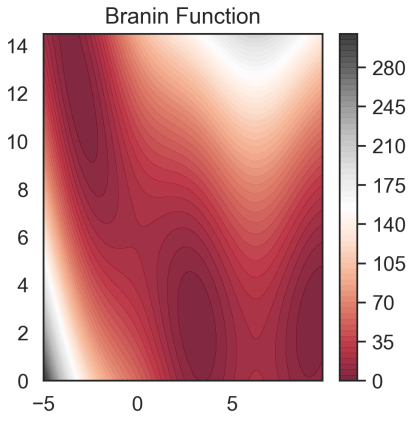
Fig. 20 Contours and surface plot of the Himmelblau function in its typical domain of definition

Branin Function

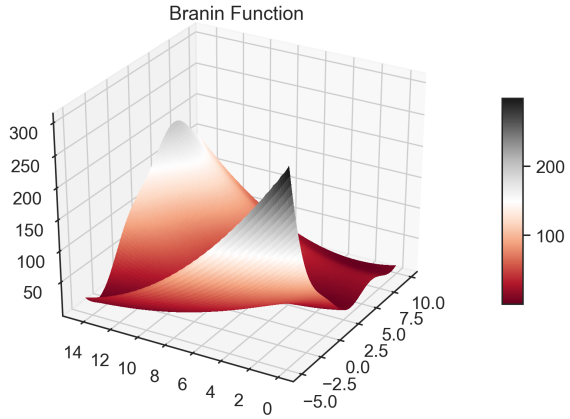
The Branin or Branin-Hoo function is another commonly used analytical function of two dimensions that is continuous and non-convex. This function is usually evaluated on the square x_1 = [-5, 10], x_2 = [0, 15]. The function has three global minima located at $[-\pi, 12.2750]$, $[\pi, 2.2750]$, and [9.4248, 2.4750]. The function contours and three dimensional surface plot are shown in Figures 21a and 21b respectively. The analytical form of this function is given by the following equation:

$$f(x, y) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t)\cos(x_1) + s \tag{7}$$

The recommended values of the parameters are: a = 1, b = $5.1/(4\pi^2)$, c = $5/\pi$, r = 6, s = 10 and t = $1/(8\pi)$.



(a) Contours of the Branin function in its domain



(b) Surface plot of the Branin function in its domain

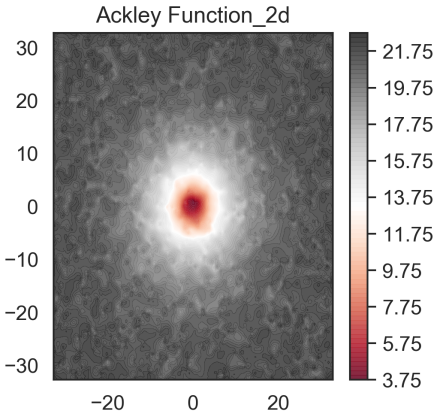
Fig. 21 Contours and surface plot of the Branin function in its typical domain of definition

Ackley Function

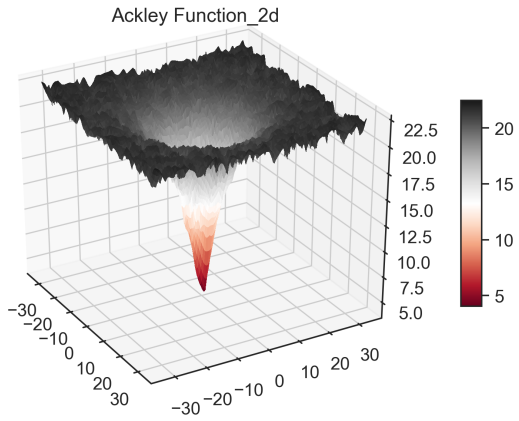
The Ackley function is widely used for testing scalability of optimization algorithms. In its two-dimensional form, it is characterized by a nearly flat outer region, and a large hole at the centre. The functional form can be used to create higher dimensional analytical problems. The function is usually evaluated on the hypercube x_i = [-32.768, 32.768]. It has a global minimum at [0, 0] and numerous local minima. The function contours and three dimensional surface plot

of the two dimensional variant of this function are shown in Figures 22a and 22b respectively. The analytical form of this function is given by the following equation:

$$f(\mathbf{x}) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1) \tag{8}$$



(a) Contours of the 2-d Ackley function in its domain



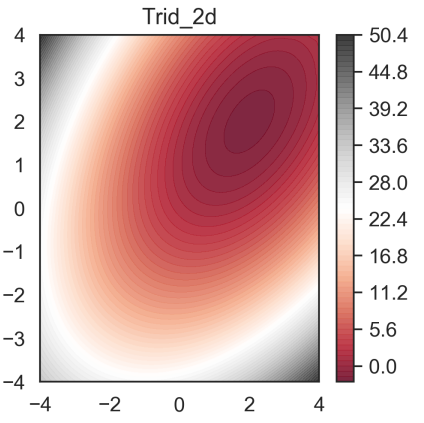
(b) Surface plot of the 2-d Ackley function in its domain

Fig. 22 Contours and surface plot of the 2-d Ackley function in its typical domain of definition

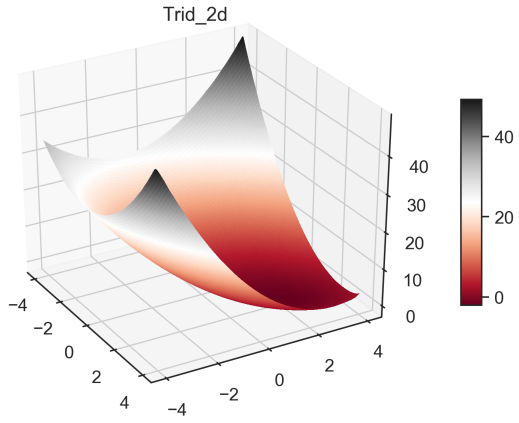
Trid Function

The Trid function is also widely used for testing scalability of optimization algorithms. In its two-dimensional form, it is a bowl-shaped function with a single global minimum at [2, 2]. The functional form can be used to create higher dimensional analytical problems. The function is usually evaluated on the hypercube $x_i = [-d^2, d^2]$. The function contours and three dimensional surface plot of the two dimensional variant of this function are shown in Figures 23a and 23b respectively. The analytical form of this function is given by the following equation:

$$f(\mathbf{x}) = \sum_{i=1}^{d}(x_i - 1)^2 - \sum_{i=2}^{d} x_i x_{i-1} \tag{9}$$



(a) Contours of the 2-d Trid function in its domain



(b) Surface plot of the 2-d Trid function in its domain

Fig. 23 Contours and surface plot of the 2-d Trid function in its typical domain of definition

Appendix: Scalability Results for Trid Function

Additional scalability results for the trid function using 100 training points are shown in Figures 24 and 25. The results follow similar trends to those shown for the Ackley canonical function shown earlier in the paper.

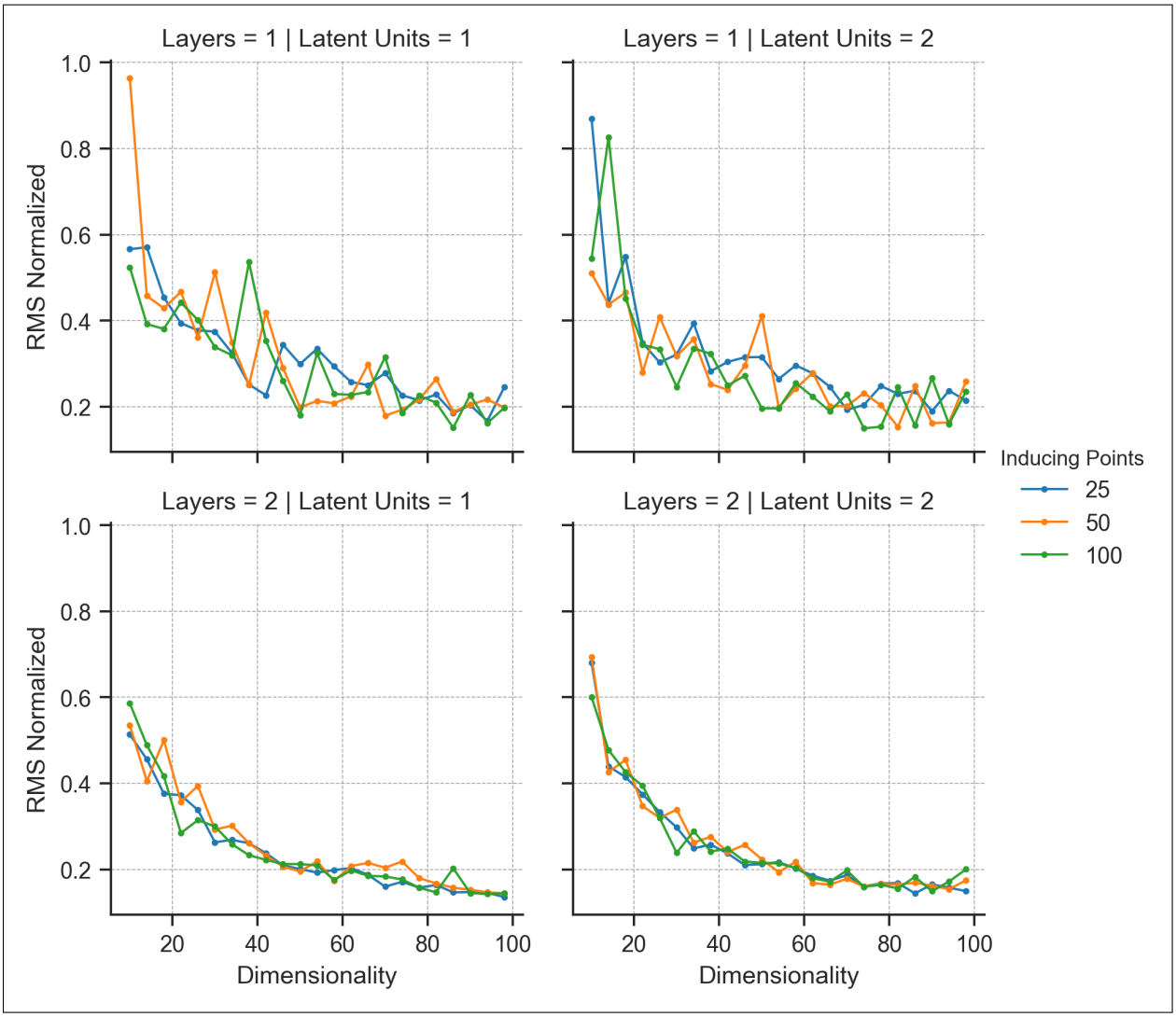


Fig. 24 Scalability for 100 training points for Trid Function

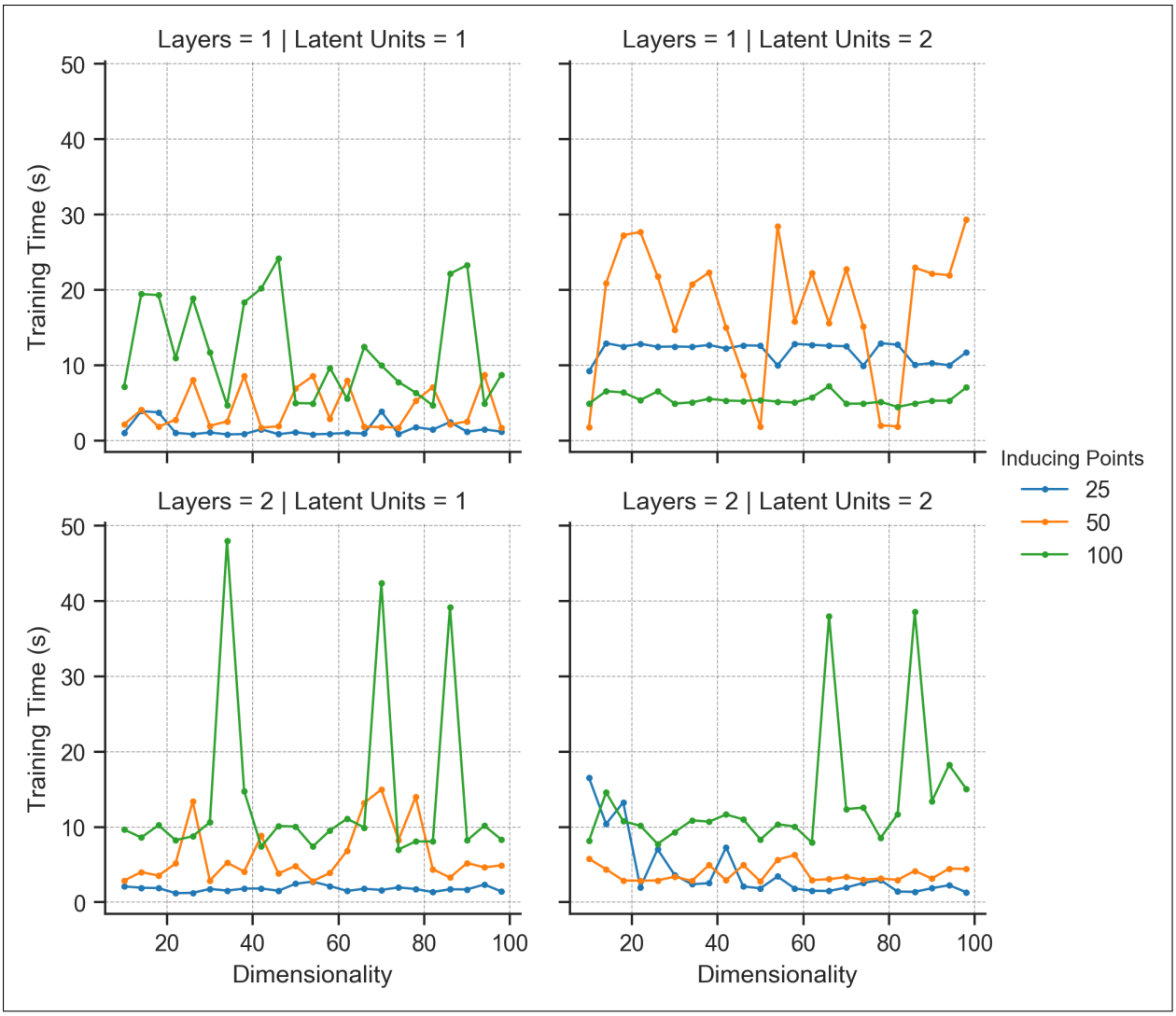


Fig. 25 Scalability for 100 training points for Trid Function